```
PPPPPPPPPPPP      LLL                IIIIIIIII    RRRRRRRRRRRR      TTTTTTTTTTTTTTT   LLL
PPPPPPPPPPPPP     LLL                IIIIIIIII    RRRRRRRRRRRRR     TTTTTTTTTTTTTTT   LLL
PPPPPPPPPPPPP     LLL                IIIIIIIII    RRRRRRRRRRRRR     TTTTTTTTTTTTTTT   LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPP          PPP  LLL                   III       RRR         RRR       TTT           LLL
PPPPPPPPPPPPP     LLL                   III       RRRRRRRRRRRRR         TTT           LLL
PPPPPPPPPPPPP     LLL                   III       RRRRRRRRRRRRR         TTT           LLL
PPPPPPPPPPPPP     LLL                   III       RRRRRRRRRRRRR         TTT           LLL
PPP               LLL                   III       RRR   RRR             TTT           LLL
PPP               LLL                   III       RRR   RRR             TTT           LLL
PPP               LLL                   III       RRR   RRR             TTT           LLL
PPP               LLL                   III       RRR      RRR          TTT           LLL
PPP               LLL                   III       RRR      RRR          TTT           LLL
PPP               LLL                   III       RRR      RRR          TTT           LLL
PPP               LLLLLLLLLLLLLLL    IIIIIIIII    RRR         RRR       TTT           LLLLLLLLLLLLLLL
PPP               LLLLLLLLLLLLLLL    IIIIIIIII    RRR         RRR       TTT           LLLLLLLLLLLLLLL
PPP               LLLLLLLLLLLLLLL    IIIIIIIII    RRR         RRR       TTT           LLLLLLLLLLLLLLL
```

```
PPPPPPPP    LL          IIIIII    PPPPPPPP    UU      UU  TTTTTTTTTT  BBBBBBBB    UU      UU  FFFFFFFFFF
PPPPPPPP    LL          IIIIII    PPPPPPPP    UU      UU  TTTTTTTTTT  BBBBBBBB    UU      UU  FFFFFFFFFF
PP      PP  LL            II      PP      PP  UU      UU      TT      BB      BB  UU      UU  FF
PP      PP  LL            II      PP      PP  UU      UU      TT      BB      BB  UU      UU  FF
PP      PP  LL            II      PP      PP  UU      UU      TT      BB      BB  UU      UU  FF
PP      PP  LL            II      PP      PP  UU      UU      TT      BB      BB  UU      UU  FF
PPPPPPPP    LL            II      PPPPPPPP    UU      UU      TT      BBBBBBBB    UU      UU  FFFFFFFF
PPPPPPPP    LL            II      PPPPPPPP    UU      UU      TT      BBBBBBBB    UU      UU  FFFFFFFF
PP          LL            II      PP          UU      UU      TT      BB      BB  UU      UU  FF
PP          LL            II      PP          UU      UU      TT      BB      BB  UU      UU  FF
PP          LL            II      PP          UU      UU      TT      BB      BB  UU      UU  FF          ....
PP          LL            II      PP          UU      UU      TT      BB      BB  UU      UU  FF          ....
PP          LLLLLLLLLL  IIIIII    PP          UUUUUUUUUU      TT      BBBBBBBB    UUUUUUUUUU  FF          ....
PP          LLLLLLLLLL  IIIIII    PP          UUUUUUUUUU      TT      BBBBBBBB    UUUUUUUUUU  FF          ....

LL            IIIIII      SSSSSSSS
LL            IIIIII      SSSSSSSS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II      SS
LL              II        SSSSSS
LL              II        SSSSSS
LL              II              SS
LL              II              SS
LL              II              SS
LL              II              SS
LLLLLLLLLL    IIIIII    SSSSSSSS
LLLLLLLLLL    IIIIII    SSSSSSSS
```

```
0000     1          .title  pli$putbuffer
0000     2          .ident  /1-005/                                        ; Edit CGN1005
0000     3                                                                  ; Edit WHM1004
0000     4                                                                  ; Edit CGN1003
0000     5                                                                  ; Edit CGN1002
0000     6
0000     7  ;
0000     8  ;*****************************************************************************
0000     9  ;*                                                                           *
0000    10  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000    11  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                    *
0000    12  ;*  ALL RIGHTS RESERVED.                                                     *
0000    13  ;*                                                                           *
0000    14  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    15  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    16  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    17  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    18  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    19  ;*  TRANSFERRED.                                                             *
0000    20  ;*                                                                           *
0000    21  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    22  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    23  ;*  CORPORATION.                                                             *
0000    24  ;*                                                                           *
0000    25  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    26  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    27  ;*                                                                           *
0000    28  ;*                                                                           *
0000    29  ;*****************************************************************************
0000    30  ;
0000    31
0000    32  ;++
0000    33  ; facility:
0000    34  ;
0000    35  ;       VAX/VMS PL1 runtime library
0000    36  ;
0000    37  ; abstract:
0000    38  ;
0000    39  ;       This module contains the pl1 runtime routines for stream output
0000    40  ;       buffer mapipulation.
0000    41  ;
0000    42  ; author: c. spitz
0000    43  ;
0000    44  ; modified:
0000    45  ;
0000    46  ;       1-002           Chip Nylander                   03-Aug-1982
0000    47  ;
0000    48  ;       Signal line-counter overflow and page-couter overflow instead of
0000    49  ;       dying in the RTL with an integer overflow.
0000    50  ;
0000    51  ;       1-003           Chip Nylander                   07-Sep-1982
0000    52  ;
0000    53  ;       Clear the top half of R2 at PLI$$PUTSKIP_R2, since some callers
0000    54  ;       do not;  fix a couple of transcription errors.
0000    55  ;
0000    56  ;
0000    57  ;       1-004   Bill Matthews   29-September-1982
```

```
                     58 ;
                     59 ;               Invoke macros $defdat and rtshare instead of $defopr and share.
                     60 ;
                     61 ;       1-005   Chip Nylander   19-December-1983
                     62 ;
                     63 ;               For PRINT files, implement PAGE by putting a formfeed in the
                     64 ;               PRN post-processing field rather than sticking a ^L in the
                     65 ;               data record.  Non-PRINT files continue to work as always.
                     66 ;
                     67 ;--
                     68
                     69 ;
                     70 ; external definitions
                     71 ;
                     72         $deffcb                         ;define file control block
                     73         $rabdef                         ;define rms rab offsets
                     74         $defstr                         ;define stream block offsets
                     75         $defpl1rtcons                   ;define runtime constants
                     76         $devdef                         ;define device bits in fab
                     77         $fabdef                         ;define fab offsets
                     78
                     79 ;
                     80 ; local data
                     81 ;
                     82
                     83         rtshare                 ;sharable
                     84
                     85 ;++
                     86 ; pli$$putnlis_r6
                     87 ;
                     88 ; functional description:
                     89 ;
                     90 ; This routine puts elements into the buffer of a pl1 stream output
                     91 ; file.  It is called by the pli$putl**** routines to copy the
                     92 ; contents of the stream block's field to the buffer.  If the string
                     93 ; in the field will not fit in the remainder of the line, and the
                     94 ; line is not empty, a skip is performed. Whether the skip was performed
                     95 ; or not, the string in field is then copied to the buffer. If the line
                     96 ; is filled, rms is called to perform a $put.
                     97 ;
                     98 ; inputs:
                     99 ;       r11 - address of the stream block
                    100 ;       ap - address of the file control block
                    101 ;       str_b_field(r11) - character string to output (in char var format)
                    102 ;       fcb_l_buf_pt - address of next available char in buf
                    103 ;       fcb_l_buf_end - address of the end of the buffer + 1
                    104 ; outputs:
                    105 ;
                    106 ; side effects:
                    107 ;
                    108 ;--
                    109         .enabl  lsb
                    110 pli$$putnlis_r6::
  2D 0C AC   07  E1 0000    111         bbc     #atr_v_print,fcb_l_attr(ap),10$ ;if print file
52  2E AC   07  AB 0005    112         bicw3   #7,fcb_w_column(ap),r2  ;round column down to last tab stop
    2E AC   52  B1 000A    113         cmpw    r2,fcb_w_column(ap)     ;already at a tab stop?
            22  13 000E    114         beql    10$                     ;if eql, yes
```

```
                52    08    A0   0010   115          addw    #8,r2                          ;round up to next tab boundary
          53    52    2E AC   A3   0013   116          subw3   fcb_w_column(ap),r2,r3        ;get number of blanks required
                2E AC   52    B0   0018   117          movw    r2,fcb_w_column(ap)            ;store column
                2A AC   52    B1   001C   118 5$:      cmpw    r2,fcb_w_linesize(ap)          ;past end of line?
                        0D    14   0020   119          bgtr    8$                             ;if gtr, yes, do a skip
    1C BC   53    20    6E    00   2C   0022   120          movc5   #0,(sp),#^x20,r3,@fcb_l_buf_pt(ap) ;insert blanks
                  1C AC   53    D0   0029   121          movl    r3,fcb_l_buf_pt(ap)            ;update buffer pointer
                        03    11   002D   122          brb     10$                            ;cont
                     00BC   30   002F   123 8$:      bsbw    pli$$putskp1_r2                ;do a skip
                  56    18 AB   3C   0032   124 10$:     movzwl  str_b_field(r11),r6            ;get length of field
                  51    1A AB   9E   0036   125 12$:     movab   str_b_field+2(r11),r1          ;get addr of start of source string
                52    56    01    C1   003A   126 15$:     addl3   #1,r6,r2                       ;get length to put (inc trailing blank)
                                   003E   127 pli$$put_contrl:
          54    2A AC   2E AC   A3   003E   128 20$:     subw3   fcb_w_column(ap),fcb_w_linesize(ap),r4 ;get length left in line
                56    54    B1   0044   129          cmpw    r4,r6                          ;enough room in line?
                        22    14   0047   130          bgtr    40$                            ;if gtr, yes
                        0A    13   0049   131          beql    30$                            ;if eql, not enuf for blank
                   2E AC    B5   004B   132          tstw    fcb_w_column(ap)               ;are we at column 0?
                        05    13   004E   133          beql    30$                            ;if eql, yes, don't skip
                     009B   30   0050   134          bsbw    pli$$putskp1_r2                ;do a skip
                        E1    11   0053   135          brb     12$                            ;try again with full line
                56    54    A2   0055   136 30$:     subw    r4,r6                          ;get length left after line is filled
             1C BC    61    54   28   0058   137          movc3   r4,(r1),@fcb_l_buf_pt(ap)      ;fill line
                  1C AC   53    D0   005D   138          movl    r3,fcb_l_buf_pt(ap)            ;update buffer position
                        51    DD   0061   139          pushl   r1                             ;save position in field
                     0088   30   0063   140          bsbw    pli$$putskp1_r2                ;put the line
                        51 8ED0   0066   141          popl    r1                             ;restore position in field
                        CF    11   0069   142          brb     15$                            ;go again
                   2E AC    52   A0   006B   143 40$:     addw    r2,fcb_w_column(ap)            ;update column
    1C BC   52    20    61    56   2C   006F   144          movc5   r6,(r1),#^x20,r2,@fcb_l_buf_pt(ap) ;copy field to buf, add blank
                  1C AC   53    D0   0076   145          movl    r3,fcb_l_buf_pt(ap)            ;update buffer pointer
                        05    007A   146          rsb                                    ;return
                              007B   147
                              007B   148          .dsabl  lsb
                              007B   149
                              007B   150 ;++
                              007B   151 ; pli$$putnedi - put the next edit field in a stream file
                              007B   152 ;
                              007B   153 ; this routine copies the vcha format string in the field into the buffer
                              007B   154 ; of a stream output file. if there is not enough room in the buffer, as
                              007B   155 ; much as will fit is copied, the buffer is written to the file and the
                              007B   156 ; remainder of the field is copied to the buffer.
                              007B   157 ;
                              007B   158 ;inputs:
                              007B   159 ; str_b_field(r11) - vcha string to output
                              007B   160 ;outputs:
                              007B   161 ; fcb_w_buf_pt(ap) is updated
                              007B   162 ; fcb_w_col(ap) is updated
                              007B   163 ; fcb_w_line(ap) may be updated
                              007B   164 ;side effects:
                              007B   165 ; r0-r6 are destroyed
                              007B   166 ;--
                              007B   167 pli$$putnedi_r6::
          51    18 AB   9E   007B   168          movab   str_b_field(r11),r1            ;get addr of field
                56    81    B0   007F   169          movw    (r1)+,r6                       ;get len of field
    52    2A AC   2E AC   A3   0082   170 10$:     subw3   fcb_w_column(ap),fcb_w_linesize(ap),r2 ;get len left in buffer
                52    56    B1   0088   171          cmpw    r6,r2                          ;field len <= buf len?
```

```
                0E   14   008B   172          bgtr    20$                             ;if gtr, then no
      1C BC     61   56   28 008D   173          movc3   r6,(r1),@fcb_l_buf_pt(ap) ;copy field to buf
           1C AC   53   D0 0092   174          movl    r3,fcb_l_buf_pt(ap)     ;update buf pointer
           2E AC   56   A0 0096   175          addw    r6,fcb_w_column(ap)     ;update column
                   05   009A   176          rsb                             ;return
                56   52   A2 009B   177 20$:     subw    r2,r6                   ;get len left in field after copy
      1C BC     61   52   28 009E   178          movc3   r2,(r1),@fcb_l_buf_pt(ap) ;copy beginning of field to buf
                51   DD 00A3   179          pushl   r1                      ;save pointer in field
           1C AC   53   D0 00A5   180          movl    r3,fcb_l_buf_pt(ap)     ;update buffer pointer
              0042   30   00A9   181          bsbw    pli$$putskp1_r2         ;write buffer to file
                51 8ED0   00AC   182          popl    r1                      ;restore pointer in field
                D1   11   00AF   183          brb     10$                     ;go again
                         00B1   184
                         00B1   185 ;++
                         00B1   186 ; pli$$put_rec - put a record to a stream file
                         00B1   187 ;
                         00B1   188 ; this routine determines the length of the record that is in the buffer,
                         00B1   189 ; and issues a $put.  upon successful completion, the column is reset to 0,
                         00B1   190 ; and the buffer is reset to the beginning of the buffer.
                         00B1   191 ;
                         00B1   192 ; calling sequence:
                         00B1   193 ;       jsb     pli$$put_rec
                         00B1   194 ;
                         00B1   195 ; inputs:
                         00B1   196 ;       fcb_l_buf(ap) - address of start of buffer
                         00B1   197 ;       fcb_l_buf_pt(ap) - address of last char to output + 1
                         00B1   198 ;       fcb_b_rab(ap) - rab
                         00B1   199 ; outputs:
                         00B1   200 ;       fcb_l_buf_pt(ap) - set to fcb_l_buf(ap)
                         00B1   201 ;       fcb_w_column(ap) - set to 0
                         00B1   202 ; side effects:
                         00B1   203 ;       r0 is destroyed
                         00B1   204 ;--
                         00B1   205
                         00B1   206 pli$$put_rec::
   0A 0C AC     17   E1 00B1   207          bbc     #atr_v_string,fcb_l_attr(ap),10$ ;if string
 50   00000000'8F   D0 00B6   208          movl    #pli$_endstring,r0      ;set end string error
                017A   31   00BD   209          brw     fail                    ;and fail
50   1C AC   14 AC   C3 00C0   210 10$:     subl3   fcb_l_buf(ap),fcb_l_buf_pt(ap),r0 ;get size of buf
     0084 CC   50   B0 00C6   211          movw    r0,<fcb_b_rab+rab$w_rsz>(ap) ;set size in rab
                         00CB   212          $put    fcb_b_rab(ap)           ;put the record
              34 AC   D4 00D5   213          clrl    fcb_l_prn(ap)           ;clr prn area
              0A 50   E8 00D8   214          blbs    r0,20$                  ;if lbs, cont
 50   00000000'8F   D0 00DB   215          movl    #pli$_rmsr,r0           ;set rms rab error
                0155   31   00E2   216          brw     fail                    ;and fail
              2E AC   B4 00E5   217 20$:     clrw    fcb_w_column(ap)        ;set column to 0
      1C AC   14 AC   D0 00E8   218          movl    fcb_l_buf(ap),fcb_l_buf_pt(ap) ;reset buffer pointer
                05   00ED   219          rsb                             ;return
                         00EE   220
                         00EE   221 ;++
                         00EE   222 ; pli$$putskip_r2 - process a skip option or format for an output stream file
                         00EE   223 ; pli$$putskp1_r2 - process one skip for an output stream file
                         00EE   224 ;
                         00EE   225 ; this routine processes the skips for an output stream file.  it differentiates
                         00EE   226 ; between print files and non-print output files. for print files, the skips are
                         00EE   227 ; accomplished by setting the number to skip in the prefix byte of the fixed
                         00EE   228 ; control area of the printer format file.  for non-print files, the skips are
```

```
                        00EE   229 ; represented by new records.  if the number to skip is 0, the file must be a
                        00EE   230 ; print file, and a carriage return is put into the prefix byte. (this is used
                        00EE   231 ; for overprinting) for print files, skip will only take you to the end of the
                        00EE   232 ; page if the number to skip is more than the number of lines left on the page.
                        00EE   233 ; for the first skip processed, the contents of the buffer are printed before
                        00EE   234 ; the buffer is cleared.
                        00EE   235 ;
                        00EE   236 ; inputs:
                        00EE   237 ;       r2 - number of skips to perform
                        00EE   238 ;       r11 - address of the stream block
                        00EE   239 ;       ap - address of the file control block
                        00EE   240 ; outputs:
                        00EE   241 ;       fcb_w_column is set to 0
                        00EE   242 ;       fcb_w_line is incremented
                        00EE   243 ;       fcb_w_page may be incremented (and the endpage condition raised)
                        00EE   244 ;
                        00EE   245 ; side effects:
                        00EE   246 ;       r0-r2 are destroyed
                        00EE   247 ;--
                        00EE   248
                        00EE   249 pli$$putskp1_r2::
          52    01    D0  00EE   250         movl    #1,r2                   ;set to skip 1
                        00F1   251 pli$$putskip_r2::
          52    52    3C  00F1   252         movzwl  r2,r2                   ;clear top half of r2
    0C 0C AC    07    E1  00F4   253         bbc     #atr_v_print,fcb_l_attr(ap),5$ ;if not print, cont
    1C AC   14 AC   D1  00F9   254         cmpl    fcb_l_buf(ap),fcb_l_buf_pt(ap) ;anything in buffer?
              05    12  00FE   255         bneq    5$                      ;if neq, cont
           34 AC    D5  0100   256         tstl    fcb_l_prn(ap)           ;anything in prn?
              03    13  0103   257         beql    6$                      ;if eql, no, don't do the put
            FFA9    30  0105   258 5$:     bsbw    pli$$put_rec            ;print the last record
           2E AC    B4  0108   259 6$:     clrw    fcb_w_column(ap)        ;reset column
              52    D5  010B   260         tstl    r2                      ;skip <= 0?
              0F    14  010D   261         bgtr    20$                     ;if gtr, then no
       07 0C AC     07  E1  010F   262         bbc     #atr_v_print,fcb_l_attr(ap),10$ ;if not print, cont
    34 AC    008D   8F  3C  0114   263         movzwl  #pli$c_cr,fcb_l_prn(ap) ;set cr in prn
              05    011A   264         rsb                             ;return
          52    01    D0  011B   265 10$:    movl    #1,r2                   ;default to skip 1
       5D 0C AC     07  E1  011E   266 20$:    bbc     #atr_v_print,fcb_l_attr(ap),30$ ;if not print, cont
          51    30 AC   3C  0123   267         movzwl  fcb_w_line(ap),r1       ;get current line number
          51    52    C0  0127   268         addl    r2,r1                   ;generate new line number
    00007FFF    8F    51  D1  012A   269         cmpl    r1,#32767               ;about to overflow?
              0A    15  0131   270         bleq    21$                     ;no - branch
    50    00000000'8F   D0  0133   271         movl    #pli$_linovrflo,r0      ;signal line overflow
                00FD  31  013A   272         brw     fail                    ;
          2C AC    51   B1  013D   273 21$:    cmpw    r1,fcb_w_pagesize(ap)   ;over a page?
              07    15  0141   274         bleq    22$                     ;if leq, then no
       2C AC    30 AC   B1  0143   275         cmpw    fcb_w_line(ap),fcb_w_pagesize(ap) ;already past pagesize?
              09    15  0148   276         bleq    25$                     ;if leq, then no
          34 AC    52   9A  014A   277 22$:    movzbl  r2,fcb_l_prn(ap)        ;set number of skips in prn
          30 AC    52   A0  014E   278         addw    r2,fcb_w_line(ap)       ;update line number
              05    0152   279         rsb                             ;return
    50    2C AC    30 AC   A3  0153   280 25$:    subw3   fcb_w_line(ap),fcb_w_pagesize(ap),r0 ;get number of lines left
              50    B6  0159   281         incw    r0                      ;on page + 1
          34 AC    50   9A  015B   282         movzbl  r0,fcb_l_prn(ap)        ;set number off skips in prn
          30 AC    50   A0  015F   283         addw    r0,fcb_w_line(ap)       ;update line number
              5C    DD  0163   284         pushl   ap                      ;push fcb addr
              00    DD  0165   285         pushl   #0                      ;set no secondary error code
```

```
                00000000'8F    DD   0167   286          pushl   #pli$_endpage            ;set endpage cond
                   0C AC  08    CA   016D   287          bicl    #atr_m_recur,fcb_l_attr(ap) ;clr recursion flag
              00000000'GF  03    FB   0171   288          calls   #3,g^pli$io_error       ;signal the condition
                   0C AC  08    C8   0178   289          bisl    #atr_m_recur,fcb_l_attr(ap) ;set recursion flag
                          05    05   017C   290          rsb                             ;return
                      FF31       30   017D   291 28$:     bsbw    pli$$put_rec            ;do a skip
                     FA 52       F5   0180   292 30$:     sobgtr  r2,28$                  ;go again as required
                          05    05   0183   293          rsb                             ;return
                                     0184   294
                                     0184   295 ;+
                                     0184   296 ;pli$$putline_r6
                                     0184   297 ; this routine process the line option or format for output stream files
                                     0184   298 ;
                                     0184   299 ;inputs:
                                     0184   300 ;       r1 - line number
                                     0184   301 ;       r11 - addr of stream block
                                     0184   302 ;       ap - addr of fcb
                                     0184   303 ;outputs:
                                     0184   304 ;       none
                                     0184   305 ;--
                                     0184   306 pli$$putline_r6::
                51    30 AC    B1   0184   307          cmpw    fcb_w_line(ap),r1       ;already at the right line?
                      08 19    19   0188   308          blss    10$                     ;if lss, then no
                      14 14    14   018A   309          bgtr    20$                     ;if gtr, then no
                   2E AC  B5    B5   018C   310          tstw    fcb_w_column(ap)        ;at col. 0?
                      0F 12    12   018F   311          bneq    20$                     ;if neq, then no
                          05    05   0191   312          rsb                             ;at right place, return [case 2.1]
                                     0192   313 10$:     ;req line > cur line
                2C AC   51    B1   0192   314          cmpw    r1,fcb_w_pagesize(ap)   ;line requested <= pagesize?
                      08 14    14   0196   315          bgtr    20$                     ;if gtr, then no
             52   51    30 AC    A3   0198   316          subw3   fcb_w_line(ap),r1,r2    ;get number of lines to skip [case 2.2]
                      FF51       31   019D   317          brw     pli$$putskip_r2         ;go to the right line
                                     01A0   318 20$:     ;req line = cur line but col ^= 0 or cur line > req line
          52   2C AC   30 AC    A3   01A0   319          subw3   fcb_w_line(ap),fcb_w_pagesize(ap),r2 ;get number of lines left
                      05 19    19   01A6   320          blss    30$                     ;if lss, then we're already over a page
                      52 B6    B6   01A8   321          incw    r2                      ;get lines left on page + 1 [case 2.3]
                      FF44       31   01AA   322          brw     pli$$putskip_r2         ;skip them
                                     01AD   323 30$:     ;brw    pli$$putpage_r6
                                     01AD   324
                                     01AD   325
                                     01AD   326 pli$$putpage_r6::
             7FFF 8F    32 AC    B1   01AD   327          cmpw    fcb_w_page(ap),#32767   ;about to overflow page counter
                      0A 19    19   01B3   328          blss    10$                     ;no
          50   00000000'8F    D0   01B5   329          movl    #pli$_pagovrflo,r0      ;yes
                      007B       31   01BC   330          brw     fail
                      55 DD    DD   01BF   331 10$:     pushl   r5                      ;save r5 in case bit src pending
             07 0C AC   07    E1   01C1   332          bbc     #atr_v_print,fcb_l_attr(ap),20$ ;branch if not print file
             35 AC   8C 8F    90   01C6   333          movb    #140,1+fcb_l_prn(ap)    ;put formfeed in field
                      10 11    11   01CB   334          brb     30$                     ;rejoin common code
                   56   01    D0   01CD   335 20$:     movl    #1,r6                   ;set size to 1
                   52   56    D0   01D0   336          movl    r6,r2                   ;set total size
                51   1A AB    9E   01D3   337          movab   <str_b_field+2>(r11),r1 ;set addr of field
                   61   0C    90   01D7   338          movb    #^x0c,(r1)              ;put formfeed in field
                      FE61       30   01DA   339          bsbw    pli$$put_contrl         ;put control char in buf
                      FED1       30   01DD   340 30$:     bsbw    pli$$put_rec            ;output record
                34 AC   01    D0   01E0   341          movl    #1,fcb_l_prn(ap)        ;set prn to one for first skip
                      32 AC    B6   01E4   342          incw    fcb_w_page(ap)          ;update page
```

```
                30 AC    01    B0  01E7   343          movw    #1,fcb_w_line(ap)       ;reset line
                         55 8ED0  01EB   344          popl    r5                      ;restore r5
                            05    01EE   345          rsb                             ;return
                                  01EF   346
                                  01EF   347 ;pli$put_end_r6
                                  01EF   348 ; inputs:
                                  01EF   349 ;        r11 - address of stream block
                                  01EF   350 ;        ap - address of file control block
                                  01EF   351 ; outputs:
                                  01EF   352 ;        none
                                  01EF   353 ; side effects:
                                  01EF   354 ;        r0-r6 are destroyed
                                  01EF   355
                                  01EF   356 pli$put_end_r6::
                   0C AC    08    C8  01EF   357          bisl    #atr_m_recur,fcb_l_attr(ap) ;set recursion flag
                21 0C AC    17    E0  01F3   358          bbs     #atr_v_string,fcb_l_attr(ap),20$ ;if put string, cont
             0F 00E6 CC     02    E1  01F8   359          bbc     #dev$v_trm,<fcb_b_fab+fab$l_dev>(ap),10$ ;if not term, cont
                0A 0C AC    07    E1  01FE   360          bbc     #atr_v_print,fcb_l_attr(ap),10$ ;if not print, cont
                                  0203   361          assume  <fcb_w_column+2> eq fcb_w_line
                      2E AC     DD  0203   362          pushl   fcb_w_column(ap)        ;save line and column
                       FEA8     30  0206   363          bsbw    pli$$put_rec            ;finish this line
                      2E AC 8ED0  0209   364          popl    fcb_w_column(ap)        ;restore them
                50  00000000'8F    D0  020D   365 10$:     movl    #ss$_continue,r0        ;set continue (needed if called from
                                  0214   366                                          ;default handler)
                   0C AC    08    CA  0214   367          bicl    #atr_m_recur,fcb_l_attr(ap) ;clr recursion flag
                            04    0218   368          ret                             ;return
                0E 0C AC    18    E1  0219   369 20$:     bbc     #atr_v_vcha,fcb_l_attr(ap),30$ ;if vcha,
                   50    14 AC    D0  021E   370          movl    fcb_l_buf(ap),r0        ;get addr of buf
                51    1C AC    50    C3  0222   371          subl3   r0,fcb_l_buf_pt(ap),r1  ;get length of string
                   FE A0    51    B0  0227   372          movw    r1,-2(r0)               ;set length
                            04    022B   373          ret                             ;return
             50    18 AC    1C AC    C3  022C   374 30$:     subl3   fcb_l_buf_pt(ap),fcb_l_buf_end(ap),r0 ;get length left in string
          1C BC    50    20    6E    00  2C  0232   375          movc5   #0,(sp),#^x20,r0,@fcb_l_buf_pt(ap) ;blank out end of string
                            04    0239   376          ret                             ;return
                                  023A   377
                                  023A   378
                                  023A   379
                   08 AC    50    D0  023A   380 fail:   movl    r0,fcb_l_error(ap)      ;set error in fcb
                            5C    DD  023E   381          pushl   ap                      ;set fcb addr
                            50    DD  0240   382          pushl   r0                      ;set error code
                      00000000'8F    DD  0242   383          pushl   #pli$_error             ;set error condition
                00000000'GF    03    FB  0248   384          calls   #3,g^pli$io_error       ;signal the condition
                            04    024F   385          ret                             ;return
                                  0250   386
                                  0250   387          .end
```

```
$$.TMP1                         = 00000001
$$.TMP2                         = 000000AC
ATR_M_RECUR                     = 00000008
ATR_V_PRINT                     = 00000007
ATR_V_STRING                    = 00000017
ATR_V_VCHA                      = 00000018
DEV$V_TRM                       = 00000002
FAB$L_DEV                       = 00000040
FAIL                              0000023A R      02
FCB_B_ENVIR                       000001C2
FCB_B_ESA                         0000012E
FCB_B_EXTRA                       0000003D
FCB_B_FAB                         000000A6
FCB_B_IDENT                       00000040
FCB_B_IDENT_NAM                   00000042
FCB_B_NAM                         000000F6
FCB_B_NUMKCBS                     0000003C
FCB_B_RAB                         00000062
FCB_C_LEN                         000001C2
FCB_C_STRLEN                      00000034
FCB_L_ATTR                        0000000C
FCB_L_BUF                         00000014
FCB_L_BUF_END                     00000018
FCB_L_BUF_PT                      0000001C
FCB_L_CNDADDR                     000001B2
FCB_L_CONDIT                      000001AE
FCB_L_DTTR                        00000010
FCB_L_ERROR                       00000008
FCB_L_KCB                         00000038
FCB_L_NEXT                        00000000
FCB_L_PREVIOUS                    00000004
FCB_L_PRN                         00000034
FCB_Q_RFA                         00000020
FCB_W_COLUMN                      0000002E
FCB_W_IDENT_LEN                   00000040
FCB_W_LINE                        00000030
FCB_W_LINESIZE                    0000002A
FCB_W_PAGE                        00000032
FCB_W_PAGESIZE                    0000002C
FCB_W_REVISION                    00000028
PLI$$PUTLINE_R6                   00000184 RG     02
PLI$$PUTNEDI_R6                   0000007B RG     02
PLI$$PUTNLIS_R6                   00000000 RG     02
PLI$$PUTPAGE_R6                   000001AD RG     02
PLI$$PUTSKIP_R2                   000000F1 RG     02
PLI$$PUTSKP1_R2                   000000EE RG     02
PLI$$PUT_CONTRL                   0000003E R      02
PLI$$PUT_REC                      000000B1 RG     02
PLI$C_CR                        = 0000008D
PLI$IO_ERROR                      ********   X    02
PLI$PUT_END_R6                    000001EF RG     02
PLI$_ENDPAGE                      ********   X    02
PLI$_ENDSTRING                    ********   X    02
PLI$_ERROR                        ********   X    02
PLI$_LINOVRFLO                    ********   X    02
PLI$_PAGOVRFLO                    ********   X    02
PLI$_RMSR                         ********   X    02
RAB$W_RSZ                       = 00000022
SIZ...                          = 00000001
SS$_CONTINUE                      ********   X    02
STR_B_FIELD                       00000018
STR_C_LEN                         00000C08
STR_L_FLD_END                     00000014
STR_L_FLD_PT                      00000010
STR_L_FP                          00000004
STR_L_FS                          0000000C
STR_L_PARENT                      00000008
STR_L_SP                          00000000
STR_L_STACK                       00000C04
STR_L_STACK_END                   00000408
SYS$POT                           ********  GX    02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000C08 (3080.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_PLI$CODE	00000250 (592.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	15	00:00:00.08	00:00:00.31
Command processing	77	00:00:00.55	00:00:02.09
Pass 1	186	00:00:06.46	00:00:12.05
Symbol table sort	0	00:00:00.62	00:00:01.14
Pass 2	75	00:00:01.42	00:00:03.03
Symbol table output	9	00:00:00.07	00:00:00.07
Psect synopsis output	2	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	364	00:00:09.23	00:00:18.72

The working set limit was 1050 pages.
34970 bytes (69 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 543 non-local and 30 local symbols.
387 source lines were read in Pass 1, producing 11 object records in Pass 2.
19 pages of virtual memory were used to define 17 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1	4
_255DUA28:[SYSLIB]STARLET.MLB;2	10
TOTALS (all libraries)	14

616 GETS were required to define 14 macros.

There were no errors, warnings or information messages.

```
MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS$:PLIPUTBUF/OBJ=OBJ$:PLIPUTBUF MSRC$:PLIPUTBUF/UPDATE=(ENH$:PLIPUTBUF)+LIB$:PLIRTM
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PLIFORMAT
LIS

PLIGETBUF
LIS

PLIMSGTXT
LIS

PLIGETEDI
LIS

PLIHEEP
LIS

PLIPUTFIL
LIS

PLIRMSBIS
LIS

PLIRECOPT
LIS

PLIOPEN
LIS

PLIREAD
LIS

PLIPROTEC
LIS

PLIREWRIT
LIS

PLIGETLIS
LIS

PLIPUTEDI
LIS

PLIPKDIVL
LIS

PLIPUTLIS
LIS

PLIMSGPTR
LIS

PLIPKDIVS
LIS

PLIPUTBUF
LIS

PLIGETFIL
LIS